

FIG. 1

### Search Criteria

Fill in at least one field. Fill more to narrow your search.  
Need high speed? Try Fast Search.

Description:	Stove
Manufacturer:	Sears
Price:	\$500 <input type="button" value="▼"/>

200

202

204

206

208

Search Now

Reset

FIG. 2

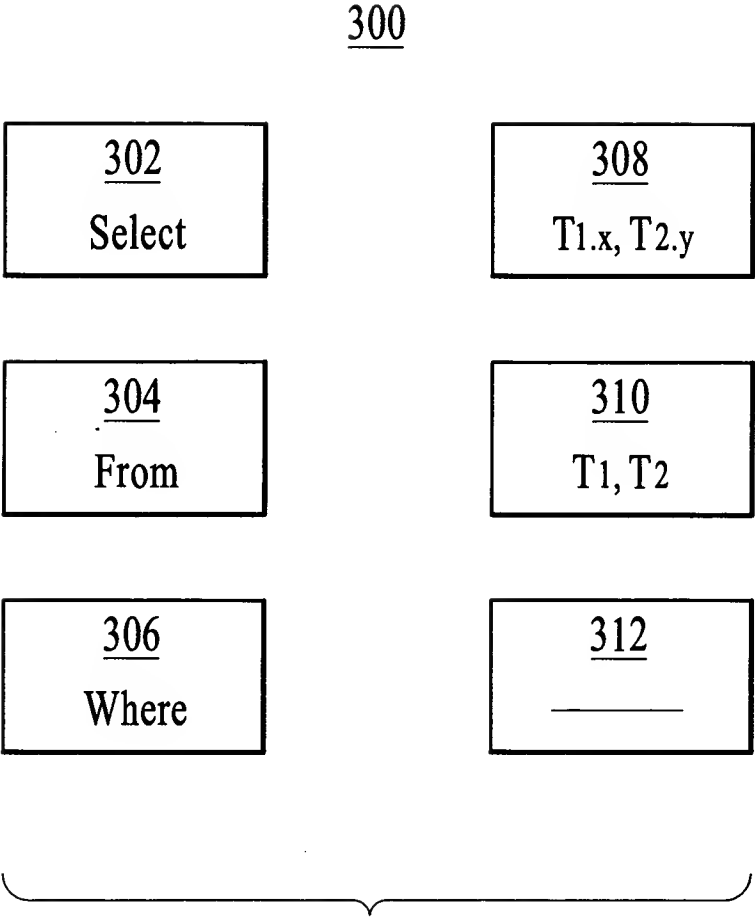


FIG. 3

Manufacturer


404

General Electric

Sears


402

Ranges

Stoves

Vacuum

Product

Stove				
Hood				

406

Sears

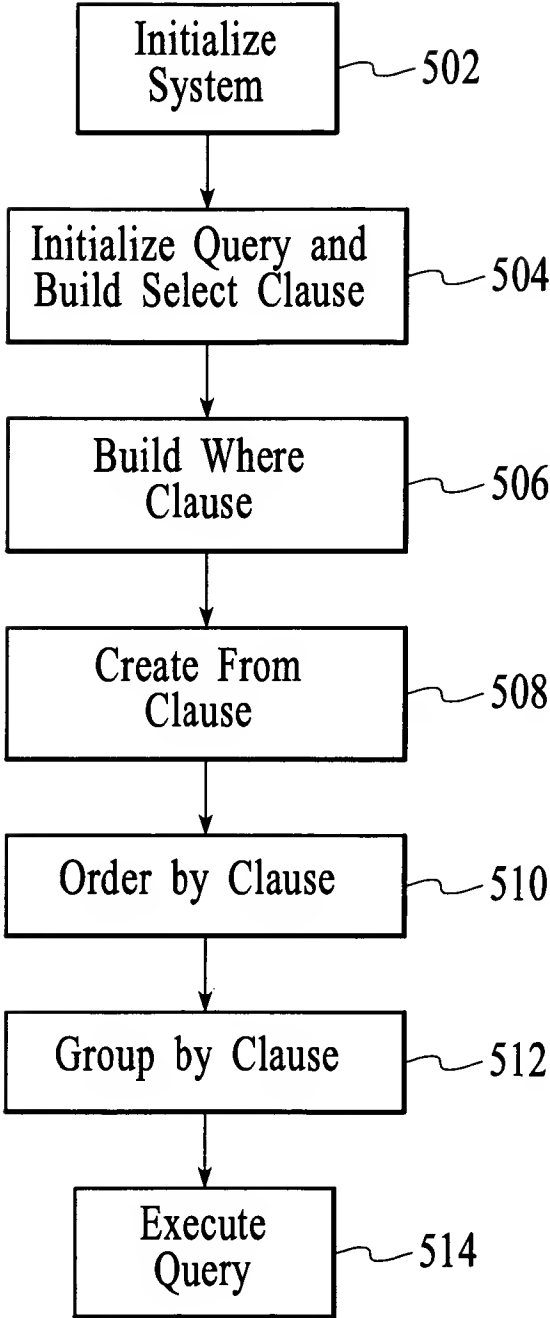
G.E.

Kenmore

G.E.

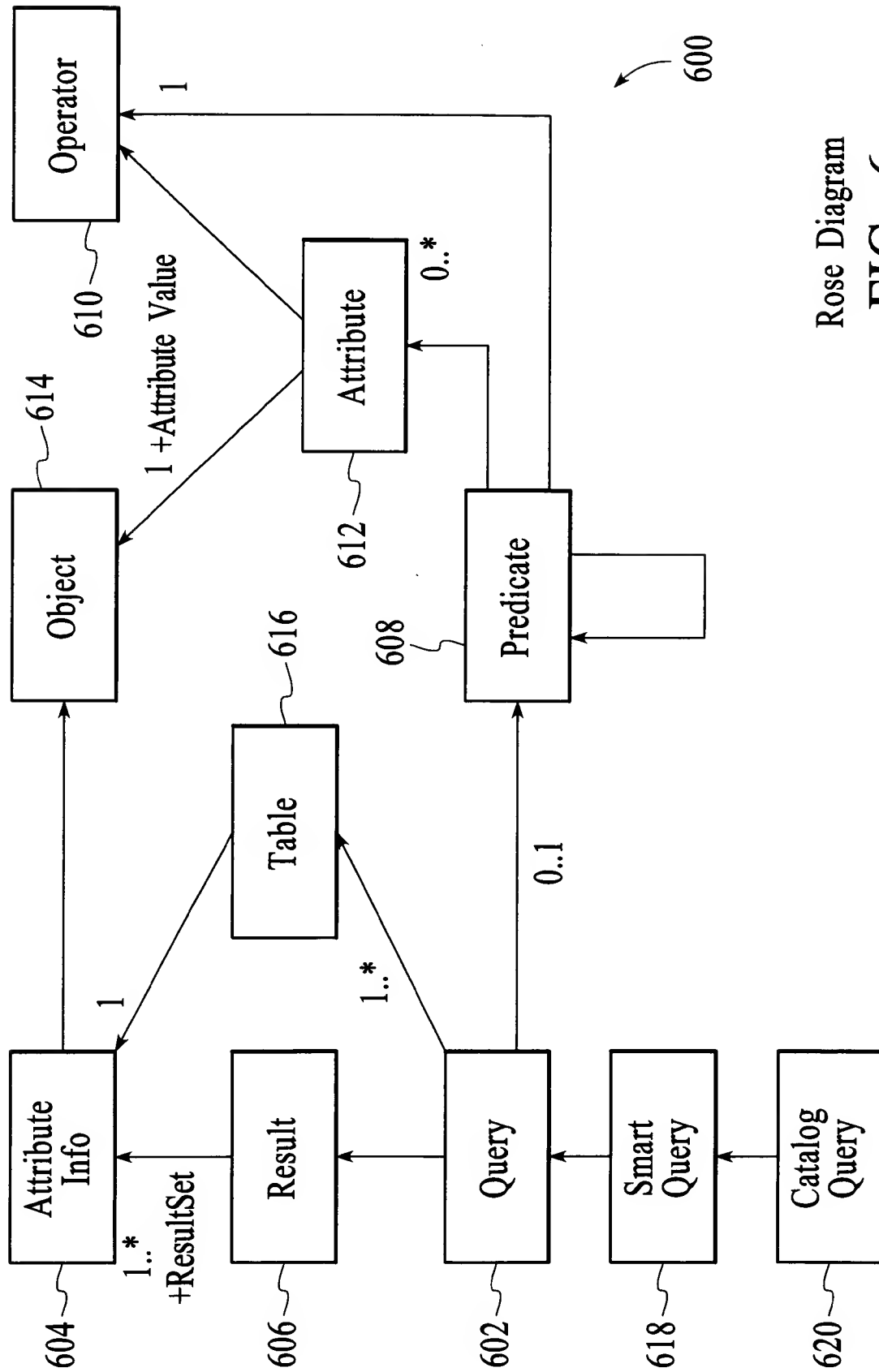
400

FIG. 4



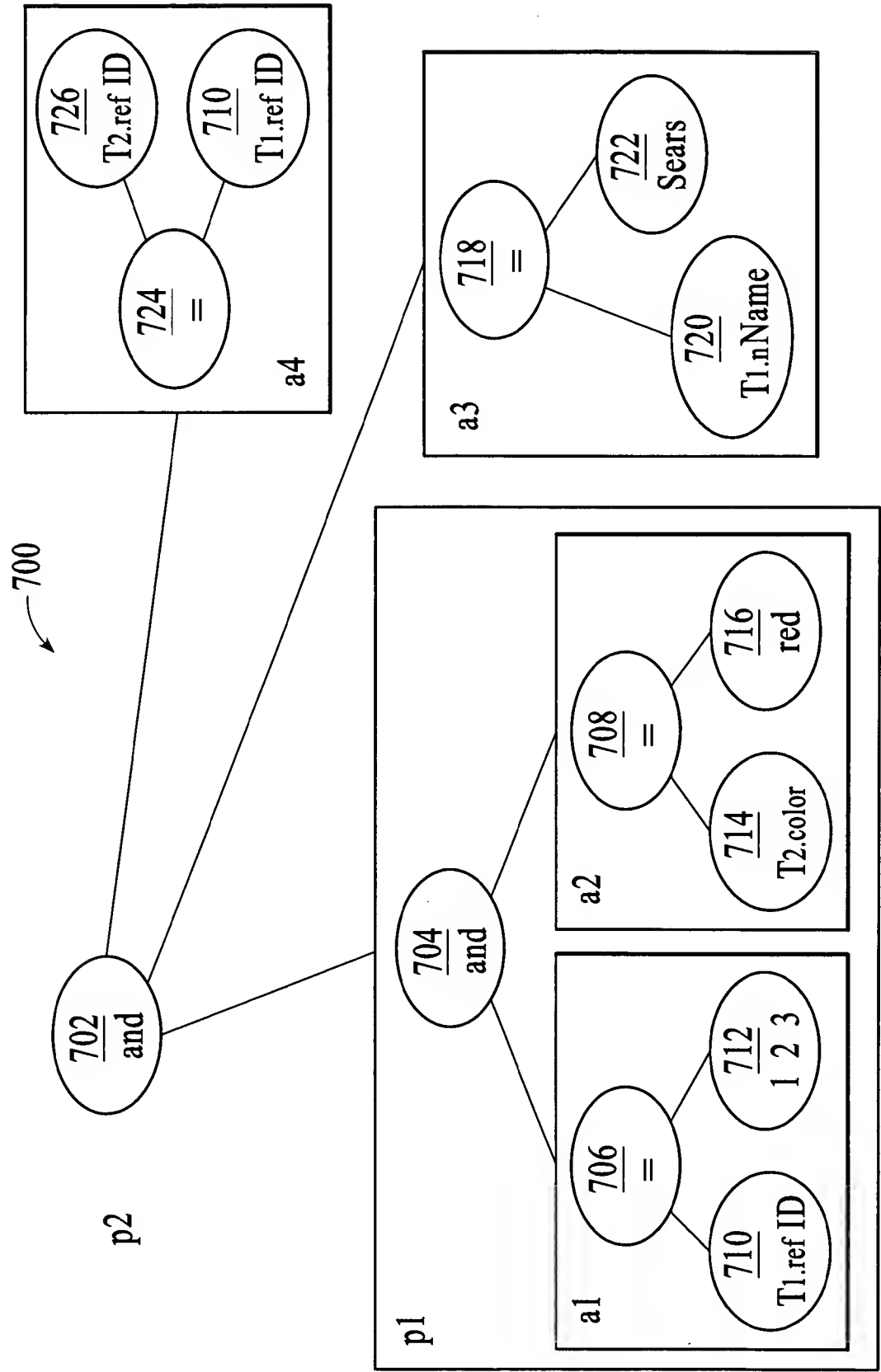
500

FIG. 5



Rose Diagram  
**FIG. 6**

FIG. 7



```
802 { // construct the simple search conditions
      Attribute attr1 = new Attribute (CatRefIdAttributeInfo, Operator.equal, "123");
      Attribute attr2 = new Attribute (ColourAttributeInfo, Operator.equal, "red");
      Attribute attr3 = new Attribute (ManufactureAttributeInfo, Operator.equal, "Sears");
      Attribute attr4 = new
      Attribute (CatRefIdAttributeInfo, Operator.equal, DescRefIdAttributeInfo,);

804 { // compose composite search conditions
      Predicate p1 = new Predicate (Operator.and, {attr1, attr2} );
      Predicate p2 = new Predicate (Operator.and, {p1, attr3, attr4} );

806 { // execute the query
      Query q = new Query ( )
      q.setResultSet ({ CatRefIdAttributeInfo, ...}) // result set contains catalog entryId
      q.setPredicate (p2);
808 { result = q.execute ( )
```

800

FIG. 8



```
public void MCQuery( ) throws Exception {  
  
    Debug.setLocalTest(true);  
  
    System.out.println(" ***** Merchant Centre ***** ");  
900 ~ CatalogQuery MCQuery = new CatalogQuery( ); ~ 901  
  
    {  
902 { // Result set  
        MCQuery.setResultSetInfo(new Result(CatEntryIdentifierAttributeInfo.getSingleton( )( ));  
        MCQuery.setResultSetInfo(new Result(StoreInvQuantityAttributeInfo.getSingleton( )( ));  
        MCQuery.setResultSetInfo(new Result(CatEntDescShortDescAttributeInfo.getSingleton( )( ));  
        MCQuery.setResultSetInfo(new Result(CatEntDescNameAttributeInfo.getSingleton( )( ));  
        MCQuery.setResultSetInfo(new Result(CatEntryTypeAttributeInfo.getSingleton( )( ));  
        MCQuery.setDistinctQualifier(true); ~ 904  
    }
```

FIG. 9A(i)

```
// Predicate set
// Part I
910a { Predicate p11 = new Predicate ( ); ~ 908a
      p11.setOperator (Operator.or);
      Attribute a111 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
      "CATEGORY X");
      a111.setUppercaseQualifier(true);
      p11.addOperand (a111);
      Attribute a112 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
      "CATEGORY10"
      a112.setUppercaseQualifier(true);
      p11.addOperand (a112);

      Predicate p12 = new Predicate ( ); ~ 908a
      p12.setOperator (Operator.and);
      p12.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p12.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p12.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
      p12.addOperand (p11);

      Predicate p13 = new Predicate ( ); ~ 908a
      p13.setOperator (Operator.and);
      p13.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p13.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p13.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
      Attribute a13 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
      "CATEGORY5");
      a13.setUppercaseQualifier(true);
      p13.addOperand (a13);

      Predicate p14 = new Predicate ( ); ~ 908a
      p14.setOperator (Operator.or);
      p14.addOperand (p12);
      p14.addOperand (p13);
```

FIG. 9A(ii)

```
// Part II
910b { Predicate p2 = new Predicate ( ); ~ 908b
      p2.setOperator (Operator.or);
      Attribute a211 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
      "CATEGORY Z");
      a211.setUppercaseQualifier(true);
      p21.addOperand (a211);
      Attribute a212 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
      "CATEGORY9"
      a212.setUppercaseQualifier(true);
      p21.addOperand (a212);

910b { Predicate p22 = new Predicate ( ); ~ 908b
      p22.setOperator (Operator.and);
      p22.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p22.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p22.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
      p22.addOperand (p21);

910b { Predicate p23 = new Predicate ( ); ~ 908b
      p23.setOperator (Operator.and);
      p23.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p23.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
      p23.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
      Attribute a23 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
      "CATEGORY4");
      a23.setUppercaseQualifier(true);
      p23.addOperand (a23);

910b { Predicate p24 = new Predicate ( ); ~ 908b
      p24.setOperator (Operator.or);
      p24.addOperand (p22);
      p24.addOperand (p23);
      p24.setNotQualifier(true);
      System.out.println(p24.toString( ));
```

FIG. 9A(iii)

```
// Part IV -- Join
Predicate p4 = new Predicate ( ); ~ 912
p4.setOperator (Operator.and);
p4.addOperand (p14);
p4.addOperand (p24);
p4.addOperand (new Attribute (StoreCEntStoreIdentifierAttributeInfo.getSingleton( ), ~ 914
Operator.eq, "2"));
p4.addOperand (new Attribute (UsersIdentifierAttributeInfo.getSingleton( ), Operator.eq,
"1001"));
//p4.addOperand (p33);

MCQuery.setPredicate(p4); ~ 916

// Join
System.out.println("Auto Join : ");
MCQuery.printJointRelationships( );

// Resolve source tables
MCQuery.resolveSourceTables( ); ~ 918

// ORDER, GROUP and HAVING set
MCQuery.addResultOrder(CatEntryIdentifierAttributeInfo.getSingleton( ),
Operator.desc); ~ 920

System.out.println("MC Query : ");
System.out.println(MCQuery.toString( ));

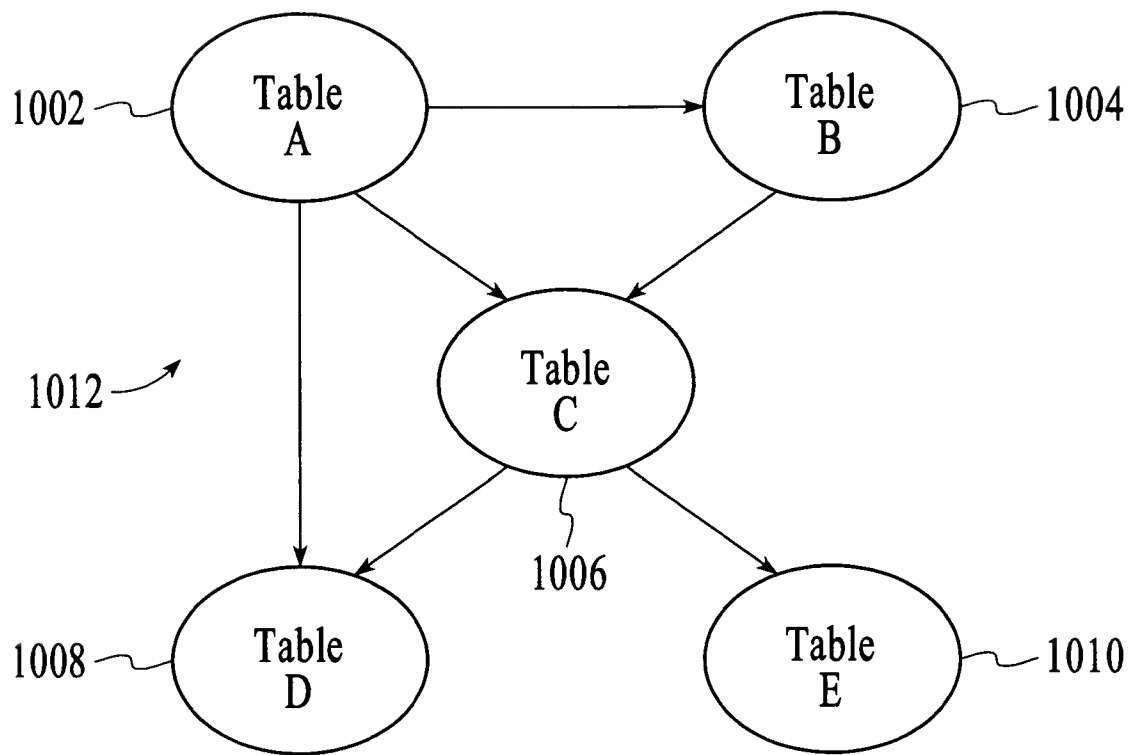
com.ibm.commerce.base.objects.Cursor cursor = new
com.ibm.commerce.base.objects.Cursor( );
java.util.Vector v = MCQuery.execute(cursor); ~ 922
System.out.println("MC Query first 10 Result: ")
System.out.println(v);
cursor.increment( );
v = MCQuery.execute(cursor); ~ 922
System.out.println("MC Query next 10 Result: ");
System.println(v);

}
```

FIG. 9A(iv)

```
public void setPredicate(Predicate aPredicate) throws Exception {  
  
    Predicate additionalP = additionalPredicate( ); ~ 924  
    if (additionalP != null) {  
        Predicate p = new Predicate( );  
        p.setOperator(Operator.and);  
        p.addOperand(aPredicate);  
        p.addOperand(additionalP);  
        setTableJointPredicate(p);  
    }  
    else  
        setTableJointPredicate(aPredicate);  
}  
926 {  
  
private void setTableJointPredicate(Predicate aPredicate) throws Exception {  
  
    Predicate jointP = resolveJointPredicate(aPredicate);  
    if (jointP != null) {  
        Predicate p = new Predicate( );  
        p.setOperator(Operator.and);  
        p.addOperand(aPredicate);  
        p.addOperand(jointP);  
        super.setPredicate(p);  
    }  
    else  
        super.setPredicate(aPredicate);  
    }  
} ~ 930
```

FIG. 9B



1000

FIG. 10

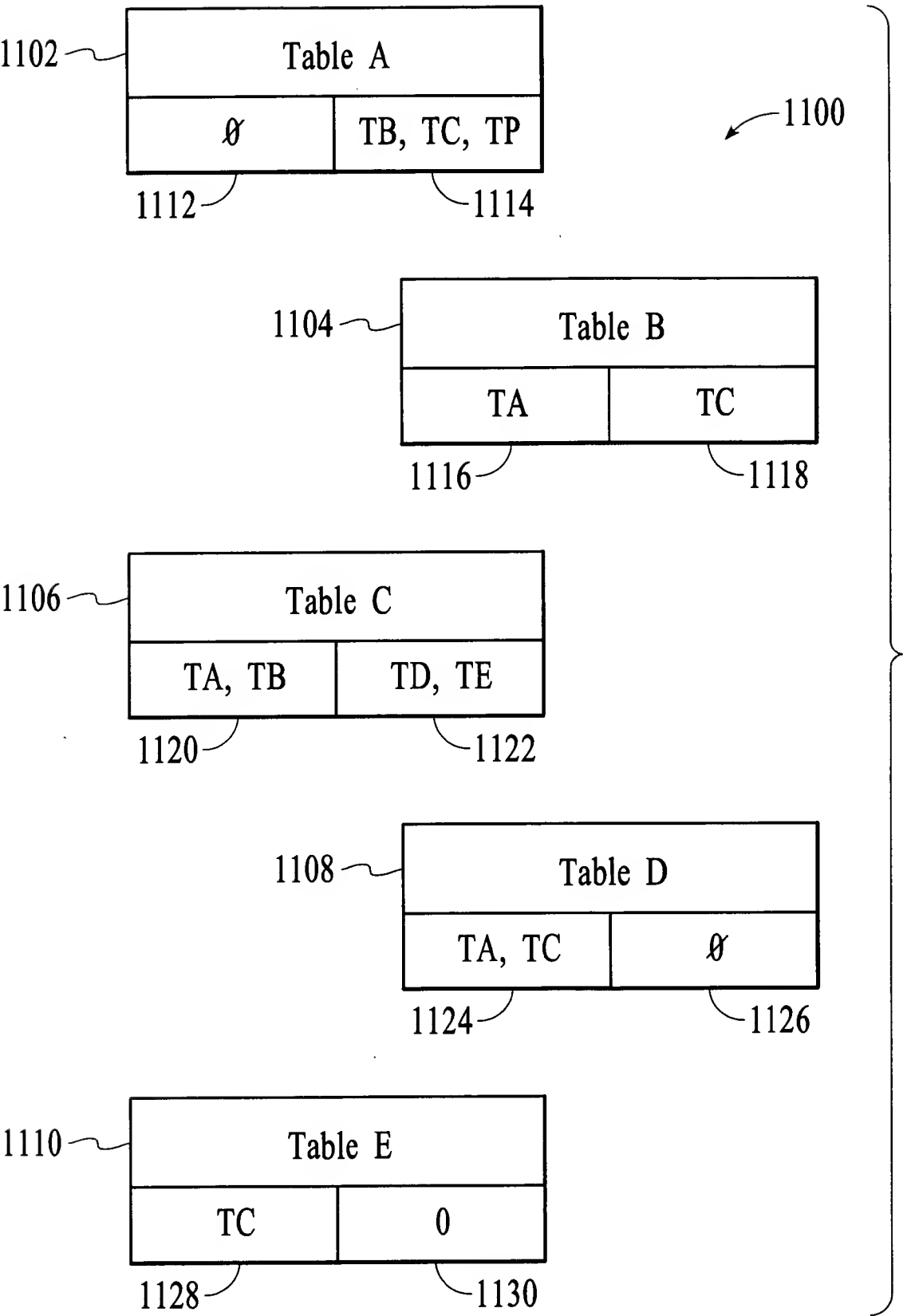


FIG. 11

FIG. 12

